Lab 5: Password-hashing (iterative hashing, salt, memory-hard functions)

Cilj vježbe je analiza korištenja hash funkcija za pohranu lozinki.

1.dio: usporedba brzih i sporih kriptografskih hash funkcija
• uspoređivali smo brzinu AES, SHA512 i SHA256
• SHA512 i i SHA256 imaju otprilike jednaka vremena izvršavanja
• jako spore hash funkcije koriste se za zaštitu vrijednih informacija

2.dio: implementacija jednostavnog sustava za autentikaciju

• stvorili smo bazu podataka korisnika i njihovih hashiranih passworda

• iako mi znamo da su neke lozinke iste, zbog soli se hashiraju u drugačije vrijednosti pa ne možemo zaključiti da su iste

• u funkciji za sign in tražimo od korisnika da uvijek unese i username i password kako napadač ne može dobiti informaciju što je od 2 unesene komponente pogrešno, ali on može zaključiti na temelju razlike u vremenu potrebnom da odgovor dođe do njega

```
import sys
from InquirerPy import inquirer
import getpass
from InquirerPy.separator import Separator

import sqlite3
from sqlite3 import Error
from passlib.hash import argon2

def verify_password(password: str, hashed_password: str) -> bool:
```

```
    # Verify that the password matches the hashed password
    return argon2.verify(password, hashed_password)

def do_sign_in_user():
    username = input("Enter your username: ")
    password = getpass.getpass("Enter your password: ")
    user = get_user(username)

    if user is None:
        print("Invalid username or password.")
        return

    password_correct = verify_password(password=password, hashed_password=user[-1])

    if not password_correct:
        print("Invalid username or password.")
        return
    print(f'Welcome "{username}".')

def get_user(username):
    try:
        conn = sqlite3.connect("users.db")
        cursor = conn.cursor()
        cursor.execute("SELECT * FROM users WHERE username = ?", (username,))
        user = cursor.fetchone()
        conn.close()
        return user
    except Error:
        return None

def register_user(username: str, password: str):
    # Hash the password using Argon2
    hashed_password = argon2.hash(password)

    # Connect to the database
    conn = sqlite3.connect("users.db")
    cursor = conn.cursor()

    # Create the table if it doesn't exist
    cursor.execute(
        "CREATE TABLE IF NOT EXISTS users (username TEXT PRIMARY KEY UNIQUE, password TEX
T)"
    )

    try:
        # Insert the new user into the table
        cursor.execute("INSERT INTO users VALUES (?, ?)", (username, hashed_password))

        # Commit the changes and close the connection
        conn.commit()
    except Error as err:
        print(err)
    conn.close()
```

```
def do_register_user():
    username = input("Enter your username: ")

    # Check if username taken
    user = get_user(username)
    if user:
        print(f'Username "{username}" not available. Please select a different name.')
        return

    password = getpass.getpass("Enter your password: ")
    register_user(username, password)
    print(f'User "{username}" successfully created.')

if __name__ == "__main__":
    REGISTER_USER = "Register a new user"
    SIGN_IN_USER = "Login"
    EXIT = "Exit"

    while True:
        selected_action = inquirer.select(
            message="Select an action:",
            choices=[Separator(), REGISTER_USER, SIGN_IN_USER, EXIT],
        ).execute()

        if selected_action == REGISTER_USER:
            do_register_user()
        elif selected_action == SIGN_IN_USER:
            do_sign_in_user()
        elif selected_action == EXIT:
            sys.exit(0)
```